DOI:
10.1109/LCOMM.2019.2911496

Document Version
Peer reviewed version

On Model Coding for Distributed Inference and Transmission in Mobile Edge Computing Systems

Jingjing Zhang and Osvaldo Simeone

***Abstract*—Consider a mobile edge computing system in which users wish to obtain the result of a linear inference operation on locally measured input data. Unlike the offloaded input data, the model weight matrix is distributed across wireless Edge Nodes (ENs). ENs have non-deterministic computing times, and they can transmit any shared computed output back to the users cooperatively. This letter investigates the potential advantages obtained by coding model information prior to ENs' storage. Through an information-theoretic analysis, it is concluded that, while generally limiting cooperation opportunities, coding is instrumental in reducing the overall computation-plus-communication latency.**

I. Introduction

Introduced by the European Telecommunications Standards Institute (ETSI), the concept of mobile edge computing is by now established as a pillar of the 5G network architecture as an enabler of computation-intensive applications on mobile devices [1]. As illustrated in Fig. 1, with mobile edge computing, users offload local data to edge servers connected to wireless Edge Nodes (ENs). These in turn carry out the necessary computations and return the desired output to the users on the wireless downlink. Most academic work on mobile edge computing has focused on the complex resource allocation problem of orchestrating computing and communication resources at the mobiles and at the ENs (see, e.g., [2] and references therein).

Papers in the line of work introduced above either assume generic applications characterized by given input-output rate requirements (e.g., [2]) or optimize the partition of the computing graph of the applications between local and edge computing. Moreover, this body of research has shown the importance of jointly designing the physical-layer transmission strategy and the computing schedule. Importantly, computing the same output at multiple ENs, while generally increasing the computation time, enables cooperation opportunities in the downlink transmission from the ENs to the users [2].

More recently, in a parallel development in the information-theoretic literature, it has been demonstrated that, if the computation of interest has specific properties, coding of either inputs or outputs can help decrease the overall latency. In particular, reference [3] demonstrated the advantages of Maximum Distance Separable (MDS) coding of input matrices in reducing the latency for distributed matrix-vector multiplication in master-worker systems. The impact of coding computational outputs was instead investigated in [4] for Map-Reduce computing tasks.

The authors are with the Department of Informatics at King's College London, UK (emails: jingjing.1.zhang@kcl.ac.uk, osvaldo.simeone@kcl.ac.uk). The authors have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement No. 725731).

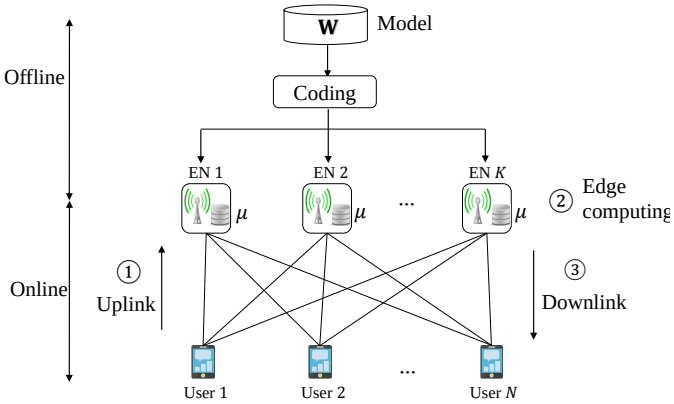

Fig. 1. Illustration of the distributed edge computing system under study.

In this letter, we investigate the role of coding in the mobile edge computing system illustrated in Fig. 1. In the system, each user wishes to compute a linear inference $\mathbf{W}\boldsymbol{x}$ on a local data vector $\boldsymbol{x}$ given a network-side model matrix $\mathbf{W}$ via offloading. The matrix $\mathbf{W}$ is generally large and hence it requires splitting across the servers of multiple ENs. Linear operations are practically important, e.g., for the implementation of recommendation systems based on collaborative filtering [5] or similarity search based on the cosine distance [6]. In both cases, the user-side data is a vector $\boldsymbol{x}$ that embeds the user profile [5] or a query [6], and the goal is to search through the matrix of all items on the basis of the inner products between the corresponding row of matrix $\mathbf{W}$ and the user-data $\boldsymbol{x}$. This letter presents an information-theoretic framework that enables the potential advantages of model coding and associated performance trade-offs to be quantified.

II. System Model and Performance Criteria

A. System Model

We consider the distributed edge computing model illustrated in Fig. 1, where N users are connected to K ENs through a shared wireless channel. For a given input vector $\boldsymbol{x} \in \mathbb{F}_{2^L}^{r\times 1}$ of rL bits provided by a user, the system aims at computing the linear inference operation $\boldsymbol{y} = \mathbf{W}\boldsymbol{x}$, where the weight, or model, matrix $\mathbf{W} \in \mathbb{F}_{2^L}^{m\times r}$ is static for a sufficiently long period of time. Each EN k can store a number of bits equivalent to a fraction $\mu \in [1/K, 1]$ of rows of matrix $\mathbf{W}$, i.e., $m\mu rL$ bits. Storage of information from matrix $\mathbf{W}$ takes place offline given the static nature of the model.

Each user n, with $n \in [N]$, has its own personal data $\boldsymbol{x}_n$, with $\boldsymbol{x}_n \in \mathbb{F}_{2^L}^{r\times 1}$ of rL bits, which is collected online

by the user, and it wishes to obtain the result of the linear operation $\boldsymbol{y}_n = \mathbf{W}\boldsymbol{x}_n$. The task is offloaded to the ENs as shown in Fig. 1. To this end, the ENs acquire the user data $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$ through uplink transmission. Second, the ENs carry out computations on the received users' data and on the stored data about $\mathbf{W}$. Finally, via downlink communication, the ENs deliver the results of the computations to the users, so that each user n can recover the required output $\boldsymbol{y}_n$.

In this letter, we make the simplifying assumption that the time needed to upload $\mathbf{X}$ to all ENs is fixed and each EN gets the entire matrix $\mathbf{X}$. This allows us to focus on the challenging problem of jointly designing offline model coding and storage at the ENs, as well as online edge computing and downlink transmission phases. The problem is formulated as follows.

Model Coding and Storage: In an *offline* phase, the model matrix $\mathbf{W}$ is linearly encoded [7] as $[\boldsymbol{c}_1^T, \cdots, \boldsymbol{c}_{m'}^T]^T = \mathbf{GW}$, where we have defined the *coding matrix* $\mathbf{G} \in \mathbb{F}_{2^L}^{m' \times m}$, with integer $m' \geq m$. Each EN k stores the subset $\mathcal{C}_k$, with $\mathcal{C}_k \subseteq \mathcal{C}$ of $|\mathcal{C}_k| \leq m\mu$ coded rows.

Edge Computing: In the *online* phase, each EN k computes inner products between all users' data received in the uplink and the available coded model rows in set $\mathcal{C}_k$. As in [8], the order in which such computations are carried out is specified by vector $\mathbf{s}_k^T = [\boldsymbol{s}_{1,k}, \cdots, \boldsymbol{s}_{m\mu,k}]$, where each element $s_{i,k} \in \mathbb{F}_{2^L}^{1\times r}$, with $i \in [m\mu]$, is selected from the set $\mathcal{C}_k$ of coded rows available at EN k. In particular, each EN k starts to compute the inner product $\boldsymbol{s}_{1,k}\mathbf{X}$ and continues computing $\boldsymbol{s}_{i,k}\mathbf{X} \in \mathbb{F}_{2^L}^{1\times N}$, for $i = 2, 3, \cdots, m\mu$. As in the literature on distributed computing, we refer to each computation $\boldsymbol{s}_{i,k}\mathbf{X}$ as an Intermediate Value (IV) [9]. A computation policy is hence defined by the coding matrix $\mathbf{G}$, *scheduling matrix* $\mathbf{S} \in \mathbb{F}_{2^{rL}}^{m\mu\times K}$, with the kth column vector given as $\mathbf{s}_k$, as well as by a *stopping criterion*, which is used by the ENs to decide when to stop the computing phase and start downlink transmission.

To formulate the stopping criterion, we define $\mathbf{m}(t) = [m_1(t), \cdots, m_K(t)]$ as the vector that indicates how many IVs have been computed by the ENs by time t, with $t = 0$ indicating the start of the computing phase and $m_k(t)$ denoting the number of computations at each EN k. Note that we have the inequalities $0 \leq m_k(t) \leq m\mu$ due to the storage constraint. We also define as

$$\mathcal{I}_k(m_k, \mathbf{s}_k) = \{\boldsymbol{s}_{i,k}\mathbf{X} : i \in [m_k]\}, \tag{1}$$

the set of first m_k IVs computed by EN k for a given choice of the scheduling vector $\mathbf{s}_k$. A computation vector $\mathbf{m}$ is said to be feasible if the union $\bigcup_{k\in[K]} \mathcal{I}_k(m_k, \mathbf{s}_k)$ of all computed IVs across all K ENs contains enough information to enable the recovery of all the outputs $\{\boldsymbol{y}_n\}_{n=1}^N$, i.e., if the conditional entropy $H\big(\{\boldsymbol{y}_n\}_{n=1}^N | \bigcup_{k\in[K]} \mathcal{I}_k(m_k, \mathbf{s}_k)\big)$ equals zero. Note that, if $\mathbf{m}$ is feasible, then any $\mathbf{m}' \geq \mathbf{m}$, where inequality is element-wise, is also feasible.

A stopping criterion for a given computation policy is defined by a set $\mathcal{M}$ of feasible computation vectors in the sense that the ENs stop computing at the first time T_C such that $\mathbf{m}(T_C)$ is in set $\mathcal{M}$, i.e.,

$$T_C = \min\{t : \mathbf{m}(t) \in \mathcal{M}\}. \tag{2}$$

As a result, the computed IVs at EN k by the end of the edge computing phase are given as $\mathcal{I}_k = \mathcal{I}_k(m_k(T_C), \mathbf{s}_k)$. As a simple example, a computation policy may require that all ENs complete all local computations, i.e., $\mathcal{M} = \{[m\mu, m\mu, \cdots, m\mu]\}$.

Downlink Communication: In this phase, the ENs send the computed IVs to the users on the downlink so that each user n can recover the desired output $\boldsymbol{y}_n$. To this end, the ENs apply conventional one-shot linear precoding as in [10], [11]. Accordingly, in each downlink transmission block, the transmitted signal at each EN $k \in [K]$ is given as $u_k = a_k s_k$, where s_k is a symbol that encodes a subset of IVs in set $\mathcal{I}_k$, and a_k is the corresponding beamforming coefficients. All the ENs that have computed the same IVs can transmit them cooperatively via joint beamforming [10], [11]. We impose the per-EN power constraint $\mathbb{E}\left[|u_k|^2\right] \leq P$. In each downlink block, the signal received by each user n is given as

$$v_n = \sum_{k=1}^{K} h_{nk}u_k + z_n, \tag{3}$$

where $h_{nk} \in \mathbb{C}$ is the channel coefficient from EN k to user n; $u_k \in \mathbb{C}$ is the defined signal transmitted by EN k; z_n is unit-power additive complex Gaussian noise. The fading channels are drawn from a continuous distribution, constant in each block, and known to all ENs.

B. Performance Analysis

As in [12], we assume that the computing time needed by each EN k to perform m_k computations is given as

$$t_k = \lambda_k + \tau m_k, \tag{4}$$

where $\lambda_k \sim \exp(\eta)$, independent across ENs, is an exponential random variable with average $1/\eta$ that models the time needed for setup at each EN k; and τ is the (deterministic) time required for each computation. Under model (4), given a stopping set $\mathcal{M}$, the random duration T_C in (2) of the computation phase can be written as the optimization

$$T_C = \max_{k\in[K]} \big(\lambda_k + \tau m_k^*(\boldsymbol{\lambda})\big), \tag{5}$$

where we have defined the stopping vector $\mathbf{m}^*(\boldsymbol{\lambda}) = [m_1^*(\boldsymbol{\lambda}), \cdots, m_K^*(\boldsymbol{\lambda})]$ for a given vector $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_K]$ as

$$\mathbf{m}^*(\boldsymbol{\lambda}) = \arg\min_{\mathbf{m}\in\mathcal{M}} \max_{k\in[K]} (\lambda_k + \tau m_k). \tag{6}$$

This follows since the time needed to realize a computation vector $\mathbf{m}$ is given by $\max_{k\in[K]}(\lambda_k + \tau m_k)$.

In the high-SNR regime of interest, we evaluate the downlink phase duration T_D by normalizing for the time $NL/\log(P)$ needed to deliver one IV, of size NL bits, to all N users, in the absence of mutual interference. Hence, the normalized communication delay δ_D is given as

$$\delta_D = \lim_{P\to\infty} \frac{T_D}{NL/\log(P)}. \tag{7}$$

For comparison, we also normalize the computation time T_C by the time τ to compute one IV for all users, obtaining the normalized computation delay $\delta_C = T_C/\tau$. Finally, the

average total normalized latency δ of the edge computing system is given as

$$\delta = \mathrm{E}[\delta_C] + \gamma \mathrm{E}[\delta_D], \tag{8}$$

where parameter γ is the ratio between the average time (in seconds) needed to compute one IV at an EN and the average time needed to transmit one IV on an interference-free channel.

III. Uncoded vs. Coded Computing

A. Uncoded Storage and Computing (UC)

Consider first a standard uncoded strategy whereby each EN stores $m\mu$ rows directly from the model matrix rows $\{\boldsymbol{w}_i\}_{i=1}^{m}$. Following, e.g., [8], the scheduling matrix $\mathbf{S}$ is designed in a cyclic manner, so that each vector $\boldsymbol{w}_i$ is repeated $K\mu$ times across all ENs. As an example, if $m = 6$, $\mu = 1/2$ and $K = 3$, then the scheduling vector are $\mathbf{s}_1 = [\boldsymbol{w}_1, \boldsymbol{w}_4, \boldsymbol{w}_5]$, $\mathbf{s}_2 = [\boldsymbol{w}_2, \boldsymbol{w}_5, \boldsymbol{w}_6]$, and $\mathbf{s}_3 = [\boldsymbol{w}_3, \boldsymbol{w}_6, \boldsymbol{w}_4]$. The stopping set $\mathcal{M}$ is defined as the set of all feasible computation vectors, so that every vector $\mathbf{m} \in \mathcal{M}$ ensures that each IV $\boldsymbol{w}_i\mathbf{X}$ has been computed by some EN.

For each IV $\boldsymbol{w}_i\mathbf{X}$ and a given feasible vector $\mathbf{m} \in \mathcal{M}$, we define as $r_i(\mathbf{m})$ the number of times that the IV has been computed across the ENs, i.e., the number of ENs whose set $\mathcal{I}_k$ contains the IV. We hence have the constraint $\sum_{i=1}^{m} r_i(\mathbf{m}) = \sum_{k=1}^{K} m_k$. To deliver a single IV computed at $r_i(\mathbf{m})$ ENs, cooperative Zero-Forcing (ZF) precoding allows $\min\{r_i(\mathbf{m}), N\}$ users to be served at the same time at the maximum high-SNR rate $\log(P)$, where $\min\{a, b\}$ represents the minimum between the two arguments a and b. This is done by choosing the precoding matrix across the $\min\{r_i(\mathbf{m}), N\}$ transmitting ENs to equal the inverse of the (square) channel matrix, upon appropriate power scaling. Hence, the normalized downlink latency (7) for this IV is given as $1/\min\{r_i(\mathbf{m}), N\}$ [10], [11]. As a result, the total latency can be characterized as follows.

Proposition 1: With the described uncoded strategy, the average total normalized latency (8) is given as

$$\delta_{UC} = \mathrm{E}\left[\frac{\max_{k\in[K]}\left(\lambda_k + \tau m_k^*(\boldsymbol{\lambda})\right)}{\tau} + \sum_{i\in[m]} \frac{\gamma}{\min\{r_i(\mathbf{m}^*(\boldsymbol{\lambda})), N\}}\right], \tag{9}$$

where the stopping vector $\mathbf{m}^*(\boldsymbol{\lambda})$ is given in (6), and the expectation is taken over the distribution of the random vector $\boldsymbol{\lambda}$.

B. MDS coded Storage and Computing (MC)

We proceed to consider an MDS-coded scheme that aims at enhancing robustness to straggling ENs [7], [9], [12]. In this scheme, the coding matrix $\mathbf{G}$ is selected as the generator matrix of an $(K\mu m, m)$ MDS code; each EN k stores $m\mu$ distinct coded rows; and the computing order at each EN is arbitrary. Furthermore, the stopping set $\mathcal{M}$ is defined such that, given the fractional cache size μ, the system waits for the fastest $\lceil 1/\mu \rceil$ ENs to finish all their computations. By definition of an $(K\mu m, m)$ MDS code, this guarantees that all the m required output elements in $\{\boldsymbol{y}_n\}_{n=1}^{N}$ can be obtained from the m IVs computed at the $\lceil 1/\mu \rceil$ ENs by treating the missing IVs from the slower $K - \lceil 1/\mu \rceil$ ENs as erasures.

With this scheme, there is no redundancy in the set of IVs computed at the ENs and hence no cooperation opportunities are available for downlink transmission. It follows that the m IVs need to be sent sequentially to each user in the downlink using orthogonal transmission, and thus the communication latency is given as $\delta_D = m$.

Proposition 2: With the described MDS coded scheme, the average total latency (8) is given as

$$\delta_{MC} = \frac{(H_K - H_{K-\lceil 1/\mu \rceil})}{\eta\tau} + m(\mu + \gamma). \tag{10}$$

Proof: Since only the fastest $\lceil 1/\mu \rceil$ ENs are required to execute their full computations, the average computation time is given as $\mathrm{E}[T_C] = \mathrm{E}[\lambda_{\lceil 1/\mu \rceil:K}] + \tau m\mu = (H_K - H_{K-\lceil 1/\mu \rceil})/\eta + \tau m\mu$, where $\lambda_{\lceil 1/\mu \rceil:K}$ is the $\lceil 1/\mu \rceil_{th}$ order statistics of exponential random variables $\{\lambda_k\}_{k=1}^{K}$, and $H_K = \sum_{k=1}^{K} 1/k$ is the K_{th} harmonic number (see [12]). ■

C. Hybrid Scheme (HS)

We now propose a hybrid scheme whose aim is to combine the robustness to stragglers afforded by the MDS-coded scheme and the cooperative downlink transmission advantages of the uncoded scheme. The proposed hybrid scheme allows the reduction in computing time via MDS coding to be traded off for savings in communication time via EN cooperation. To this end, we concatenate an $(\rho_1 m, m)$ MDS code for some $\rho_1 \geq 1$ with a repetition code that replicates each coded vector to ρ_2 ENs. Controlling the design parameters (ρ_1, ρ_2), the scheme ranges from uncoded storage $(\rho_1 = 1)$ to MDS coding $(\rho_2 = 1)$.

More precisely, following [7], in order to ensure an even distribution of coded rows, the $\rho_1 m$ coded rows $\{\boldsymbol{c}_i\}_{i=1}^{\rho_1 m}$ are split into $\binom{K}{\rho_2}$ disjoint subsets. Each subset $\mathcal{C}_{\mathcal{K}}$ consists of $b = (\rho_1 m)/\binom{K}{\rho_2}$ coded rows, and is indexed by a subset $\mathcal{K} \subseteq [K]$ of size ρ_2, i.e., $|\mathcal{K}| = \rho_2$. Each EN k stores all the rows in the set $\bigcup_{\mathcal{K}:k\in\mathcal{K}} \mathcal{C}_{\mathcal{K}}$, with cardinality $b\binom{K-1}{\rho_2-1} = \rho_1\rho_2 m/K$. Due to the storage constraint $m\mu$ at each EN, we have the constraint

$$\rho_1\rho_2 \leq K\mu. \tag{11}$$

We select the stopping set in a manner similar to the MDS coded strategy, so that the computing phase is completed as soon as q ENs complete all their computations, where q is a design parameter. Following [7, Proposition 1], the three design parameters (q, ρ_1, ρ_2) need to satisfy the constraint

$$\binom{K}{\rho_2} - \binom{K-q}{\rho_2} \geq \frac{1}{\rho_1}\binom{K}{\rho_2} \tag{12}$$

in order to ensure that m distinct coded IVs are computed across the ENs and hence all desired outputs can be recovered. It can be observed that the choice of parameters (ρ_1, ρ_2) depends on system parameters K, μ and γ, which are constant, and design parameter q. These parameters are expected to be constant for long periods of time and hence frequent re-encoding is not necessary.

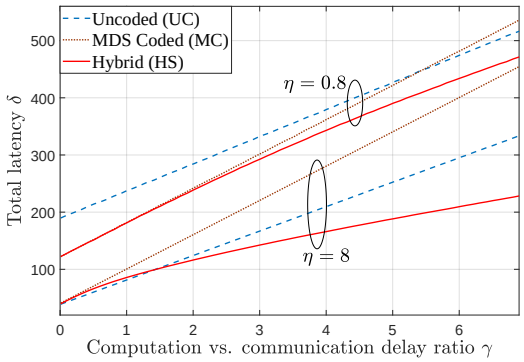

Fig. 2. Latencies of UC, MC and HS versus ratio r for $K = N = 6$, $\tau = 0.005$, $m = 60$, $\mu = 0.5$ and different values of η.

At the end of the computing phase, each computed IV $\boldsymbol{c}_i\mathbf{X}$ is available at r_i ENs, where r_i can be shown to lie in the interval $[r_{min}, r_{max}]$, with $r_{min} = \max\{\rho_2 - (K-q), 1\}$ and $r_{max} = \min\{q, \rho_2\}$ in a manner similar to [7]. Moreover, for any $r_i \in [r_{min}, r_{max}]$, the number of computed IVs is $B_i = \binom{q}{r_i}\binom{K-q}{\rho_2 - r_i} b$ since there are $\binom{q}{r_i}\binom{K-q}{\rho_2-r_i}$ subsets of ENs that have computed the same IVs. For downlink transmission, in order to maximizing cooperative opportunities, the computed IVs are sent in descending order of redundancy r_i by using cooperative ZF precoding to serve r_i users simultaneously.

Proposition 3: With the described hybrid scheme, the average total latency (8) is given as

$$\delta_{HS} = \min_{q} \left[\frac{(H_K - H_{K-q})}{\eta\tau} + m\mu + \gamma \min_{(\rho_1,\rho_2)} \left(\sum_{r_i=r_q}^{r_{max}} \frac{B_i}{r_i} + \frac{m - \sum_{r_i=r_q}^{r_{max}} B_i}{r_q - 1} \right) \right], \quad (13)$$

where we have defined $r_q = \inf\left\{r : \sum_{r_i=r}^{r_{max}} B_i \leq m\right\}$; and the optimization over parameters $q \in [\lceil 1/\mu \rceil, K]$, $\rho_1 \in [1, (q+1)/q, \cdots, K/q]$, and $\rho_2 \in [\lfloor q\mu \rfloor : \lfloor K\mu \rfloor]$ is constrained by Condition (11) and (12).

Proof: Given any design parameter $q \in [\lceil 1/\mu \rceil, K]$, the average computation time is evaluated as in Proposition 2, with the computing latency given as $(H_K - H_{K-q})/(\eta\tau) + m\mu$ in (10). Using downlink transmission, the B_i IVs with redundancy r_i require a communication latency B_i/r_i using cooperative ZF as explained in Section III-A. In order to deliver m IVs, the IVs with redundancy $r_i \in [r_q, r_{max}]$ are sent in full, while only $m - \sum_{i=r_q}^{r_{max}} B_i$ IVs with redundancy $r_q - 1$ need to be delivered. The corresponding total communication latency is optimized over all design parameters (q, ρ_1, ρ_2) that satisfy Condition (11) and (12). ∎

IV. Example and Discussion

In this section, we present a numerical example for a system with $K = N = 6$ ENs and users, $m = 60$ row vectors in model matrix $\mathbf{W}$, and fractional cache size $\mu = 0.5$. We also set the per-IV computation time to $\tau = 0.005$ and the average set-up time to different values of $1/\eta$. In Fig. 2, we plot the overall average latency δ as a function of the ratio γ between normalized computation and communication times.

As seen in Fig. 2, as γ increases, the total latencies of both UC in (9) and MC in (10) grow linearly, and the relative performance depends on the values of γ and η. When η is small, i.e., $\eta = 0.8$, the variability in the computing times of the ENs is high, and MDS coding for the most part outperforms the UC scheme due to its robustness to stragglers. This is unless γ is large enough, in which downlink transmission latency becomes dominant and the UC scheme can benefit from redundant computations via cooperative EN communication. In contrast, for larger values of η, the computing times have low variability and MDS coding is uniformly outperformed by the UC scheme.

We also observe that the proposed hybrid coding strategy is effective in trading off computation and communication latencies by controlling the balance between robustness to stragglers and cooperative opportunities via the design of parameters (q, ρ_1, ρ_2). In fact, by increasing q and ρ_2, this approach can decrease the communication latency at the cost of a larger computing latency. Apart from very small values of γ for large η, the scheme is seem to outperform both MDS and UC strategies.

An interesting open problem is to design a hybrid strategy that generalizes both the proposed MDS and UC schemes by properly optimizing the scheduling matrix in a manner akin to UC. Other aspects that are left for future work include the investigation of coding schemes that enable the use of ENs' partial computations [12]; of transmission strategies that carry out simultaneous edge computing and downlink communications; of the impact of partial uplink connectivity; and of protocols able to accommodate an arbitrary number of computing tasks.

References

[1] T. Taleb and et al, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, May 2017.

[2] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, June 2015.

[3] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, March 2018.

[4] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Magazine*, vol. 55, no. 4, pp. 34–40, April 2017.

[5] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*. Springer Berlin/Heidelberg, 2007, pp. 291–324.

[6] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," in *WWW*, 2007, pp. 131–140.

[7] J. Zhang and O. Simeone, "Improved latency-communication trade-off for map-shuffle-reduce systems with stragglers." [Online]. Available: http://arxiv.org/abs/1808.06583

[8] E. Ozfatura, S. Ulukus, and D. Gündüz, "Distributed gradient descent with coded partial gradient computations." [Online]. Available: https://arxiv.org/abs/1811.09271

[9] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *IEEE Globecom Workshops (GC Workshop)*, Dec 2016, pp. 1–6.

[10] J. Zhang and O. Simeone, "Fundamental limits of cloud and cache-aided interference management with multi-antenna edge nodes." [Online]. Available: http://arxiv.org/abs/1712.04266

[11] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3092–3107, May 2017.

[12] A. Mallick, M. Chaudhari, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication." [Online]. Available: http://arxiv.org/abs/1804.10331